Android 官方开发教程中文版

创建你的第一个 App

创建你的第一个 App

欢迎加入 Android 应用开发。

本课程将教你如何创建你的首个 Android 应用。你将学习到如何创建 Android 项目和运行它的可调试版本。你还会学习到 Android 应用设计的基本原理，包括如何创建简单用户界面和处理用户输入。

在你开始本课程之前，你要确信你的开发环境已经设置好，你需要：

1. 下载 Android SDK。
2. 为 Eclipse 安装 ADT 插件（如果你的 IDE 是 Eclipse）。
3. 使用 SDK Manager 下载最新的 SDK 工具和平台。

如果你还没有完成这些任务，那么开始下载 Android SDK 并完成随后的安装步骤。一旦你完成安装，也就做好了开始本课程的准备工作。

本课程采用教程的格式，通过逐步构建一个小型的 Android 应用，教你了解一些关于 Android 开发的基本概念，因此，你应该遵循教程中的每一个步骤。

创建 Android 项目

一个 Android 项目包含 Android 应用的所有文件以及源代码。Android SDK 工具可以很轻松地使用默认的目录和文件开始一个新的 Android 项目。

本课程将演示如何使用 Eclipse（含 ADT 插件）或 SDK 工具从命令行创建新项目。

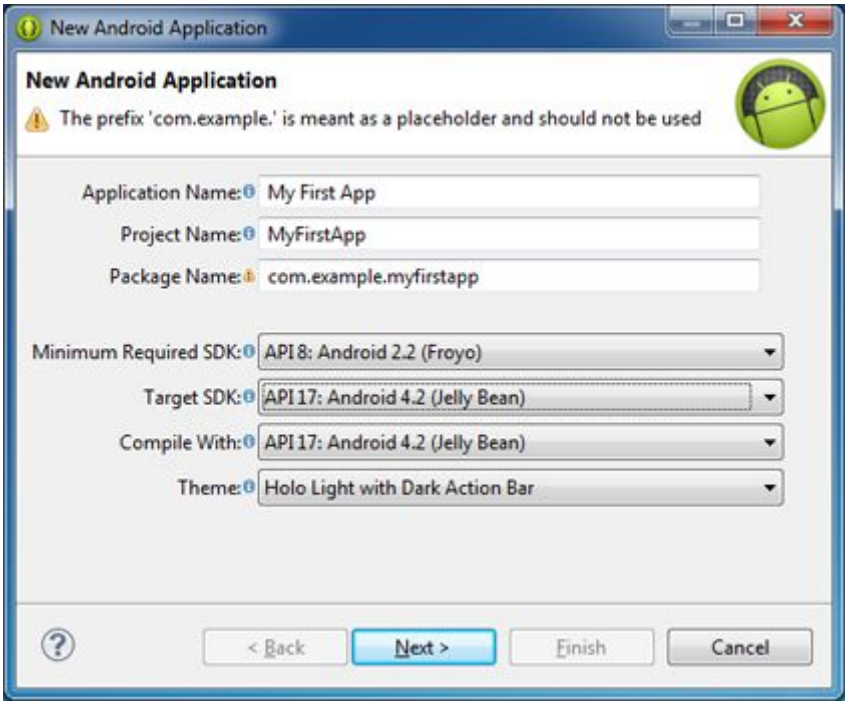
注意：*你应该已经安装了 Android SDK，并且如果你使用的是 Eclipse，那么你也应该安装了 ADT 插件（版本 21.0.0 或更高），如果你还没有完成这些，在开始本课程之前请按照前面的指南安装 Android SDK。*

使用 Eclipse 创建项目

1. 在工具样上点击 **New**  。
2. 在出现的窗口中，打开 **Android** 文件夹，选择 **Android Application Project**，点击 **Next**。
3. 填写以下内容：
   - **Application Name**：用户看到的应用名称，在本项目中，使用“My First App”。
   - **Project Name**：项目目录的名称，并且在 Eclipse 中也能看到的名称。
   - **Package Name**：应用的包名（遵循 Java 程序语言中包的命名规则）。你的包名在所有安装到 Android 系统的包中必须是唯一的，因此，包名最好是你的组织或出版商的域名反转。在本项目中，你可以使用类似“com.example.myfirstapp”的包名。无论如何，你不能使用“com.example”的包命在谷歌发布你的应用。
   - **Minimum Required SDK**：你的应用支持的 Android 系统的最低版本和它使用的 API 级别。为了支持尽可能多的设备，你应该将它设置为所允许的最低版本以允许你的应用提供所有核心功能。如果你的应用中某个功能只能在新版本的 Android 系统中运行，并且它不是核心功能的关键部分，你可以启用仅在支持

的版本上运行该功能(我们将在“**支持不同平台版本**”中进一步讨论)。本项目中使用默认值。

- **Target SDK**：表示用来测试你的应用的 Android 系统的最高版本和 API 级别。当新的 Android 系统可用时，你应该在新版本上测试你的应用并更新这个值以匹配最新的 API 级别，从而利用新的平台特性。
- **Compile With**：用以编译你的应用的平台版本。默认情况下，这会设置为你的 SDK 中可用的最新版本的 Android 系统（应该是 Android4.1 或更高版本，如果你没这样一个可用版本，你必须用 SDK Manager 安装一个）。你仍然可以创建支持旧版本的应用，但把构建目标设置为最新版本可以允许你启用新功能并且在最新设备上优化你的应用以便为用户提供更好的用户体验。
- **Theme**：你的应用使用的 Android 界面风格。你可以保持默认值。



图一 Eclipse 的创建新 Android 项目的向导

点击 Next。

4. 在配置项目的下一个屏幕中，使用默认选项，点击 **Next**。
5. 下一个屏幕可以帮助你为你的应用创建一个执行图标。
   你可以用多种方式自定义一个图标，用工具为不同屏幕分辨率生成图标。在发布你的应用之前，你应该确保你的图标符合设计指南定义的规范。
   点击 **Next**。
6. 现在你可以选择一个 Activity 模板来创建你的应用。本项目中，选择 **BlankActivity** 并且点击 **Next**。
7. 为 Activity 保留默认状态，点击 **Finish**。

你的 Android 项目现在设置了默认文件，准备开始创建应用吧。

使用命令行工具创建 Android 项目

如果你没有使用带 ADT 插件的 Eclipse，你也可以从命令行使用 SDK 工具来创建 Android 项目。

1. 改变当前路径到 Android SDK 的 tools 目录下。

2. 执行：

```
android list targets
```

这将列出你的 SDK 已经下载的 Android 平台，找到你想编译应用的平台，注意目标 ID。我们建议你选择所允许的最高版本，虽然你可以构建支持旧版本的应用，但设定为最新版本可以为最新设备优化你的应用。

如果你没有看到任何目标列表，你需要使用 SDK Manager 安装一些平台。

3. 执行：

```
android create project --target <target-id> --name MyFirstApp \
--path <path-to-workspace>/MyFirstApp --activity MainActivity \
--package com.example.myfirstapp
```

用上面得到的目标 ID 替换<target-id>，用你想保存 Android 项目的本地路径替换<path-to-workspace>。

运行你的 App

如果你按照上一节课创建了 Android 项目，它包含了一个默认的“Hello World”源文件，允许你立即运行该应用。

如何运行你的应用依赖两件事：你是否有一个真实的 Android 设备；你是否在使用 Eclipse。本节将演示如何在一个真实的 Android 设备或 Android 模拟器上安装和运行你的应用，以及分别在两种情况下运行：Eclipse 和命令行工具。

在运行你的应用之前，我们应该了解一些有关 Android 项目的目录和文件。

AndroidManifest.xml

清单文件描述了应用的基本特征并定义了它的每个组件，随着课程的深入，你将了解这个文件中的各种定义。

清单文件中最重要的元素之一是<uses-sdk>元素，它用 android:minSdkVersiont 和 android:targetSdkVersion 两个属性定义了你的应用兼容不同的 Android 版本。在你的首个 App 中，它看起来类似这样：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" ... >
    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="17" />
    ...
</manifest>
```

你应该把 android:targetSdkVersion 设置的尽可能高并在相应的平台上测试你的应用。更多信息请参见“**支持不同平台版本**”。

src/

应用的主要源文件的存放目录。默认情况下，它包含一个当通过应用图标启动应用时要运行的 Activity 类。

res/

包含了几个和应用资源相关的子目录，有这么几个：

drawable-hdpi/

存放为高分辨率屏幕设计的可绘制对象（如位图）。其它以 drawable 开头的目录存放为其它分辨率设计的资源。

layout/

存放定义应用的用户界面的文件。

values/

存放包含其它资源集合的各种 XML 文件，如字符串和颜色的定义。

当你构建和运行默认 Android 应用时，默认的 Activity 类会启动并加载布局文件显示“Hello World”，结果虽然平平淡淡，但重要的是，在你开发之前了解了如何运行你的应用。

在真实设备上运行

如果你有一部真实的 Android 设备，下面将告诉你如何安装和运行你的应用。

1. 用 USB 连线连接你的设备和开发机器，如果你在 Windows 下开发，你可能需要为你的设备安装合适的 USB 驱动程序。要获得安装驱动程序的帮助，请参看 OEM USB Drivers 文档。
2. 在你的设备上启用 USB debugging
   - 大多数设备运行 Android3.2 或更旧的版本，你可以在 **Settings>Applications>Development** 下找到这个选项。
   - 在 Android4.0 或更新版本中，这个选项在 Settings>Developer options 中。
     注意：*在 Android4.2 或更新版本中，**Developer options** 默认是隐藏的，要让它可见，进入 **Settings>About phone** 并且轻触 **Build number** 七次，返回上一级屏幕就可以找到 **Developer options**。*

从 Eclipse 中运行：

1. 打开的你项目中的一个文件，并在工具栏上点击运行。
2. 在 **Run as** 窗口中，选择 **Android Application** 然后点击 **OK**。

Eclipse 将把应用安装到你连接的设备上并启动它。

你也可以从命令行运行你的应用

1. 把路径改变到你的 Android 项目的根目录中，执行：
   ant debug
2. 确保你的 Android SDK platform-tools/目录在你的 PATH 环境变量中，执行：
   adb install bin/MyFirstApp-debug.apk
3. 在你的设备上找到 MyFirstActivity 并打开它。

这就是如何构建和在设备上运行你的应用，要进入开发，让我们继续下一节课。

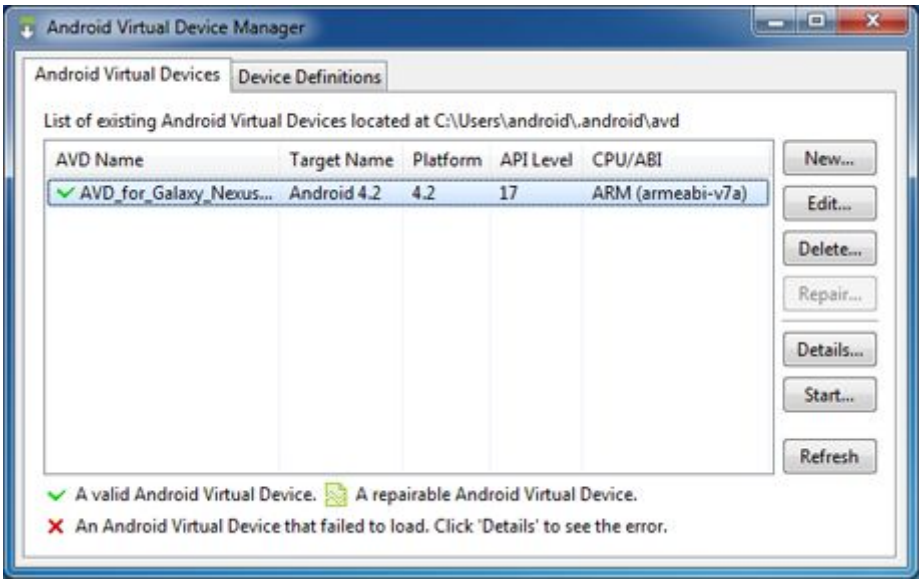
在模拟器上运行

无论你是使用 Eclipse 还是命令行，要在模拟器上运行你的应用，首先要创建一个 Android 虚拟设备（AVD）。一个 AVD 是 Android 模拟器的设备配置，允许你模拟不能的设备。

要创建 AVD：

1. 执行 Android 虚拟设备管理工具
   a. 在 Eclipse 中，点击工具栏的 Android Virtual Device Manager。
   b. 在命令行中，把路径改变到<sdk>/tools/并且执行：
      androiv avd
2. 在 Android 虚拟设备管理工具中，点击 **New**。
3. 填写 AVD 的详细信息，给它一个名字，目标平台，SD 卡的大小以及皮肤（默认为 HVGA）。
4. 点击 **Create AVD**。
5. 在 Android 虚拟设备管理工具中选择新的 AVD，点击 **Start**。

6. 模拟器启动之后，解锁模拟器的屏幕。



从 Eclipse 中运行：

1. 打开的你项目中的一个文件，并在工具栏上点击运行。
2. 在 **Run as** 窗口中，选择 **Android Application** 然后点击 **OK**。

Eclipse 将把应用安装到模拟器上并启动它。

你也可以从命令行运行你的应用

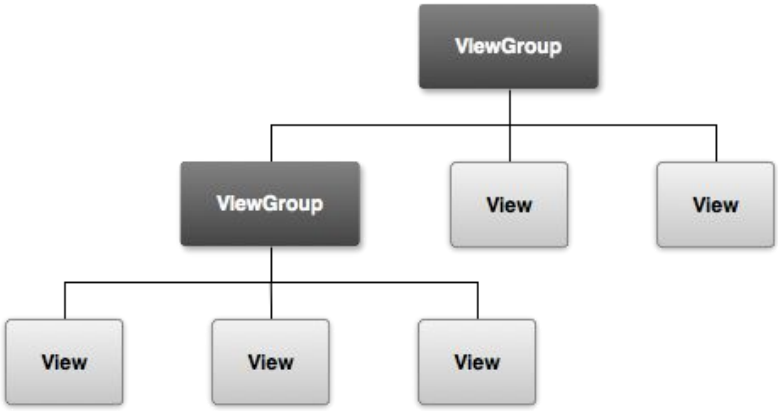
4. 把路径改变到你的 Android 项目的根目录中，执行：
   ant debug
5. 确保你的 Android SDK platform-tools/目录在你的 PATH 环境变量中，执行：
   adb install bin/MyFirstApp-debug.apk
6. 在模拟器中找到 MyFirstActivity 并打开它。

这就是如何构建和在设备上运行你的应用，要进入开发，让我们继续下一节课。

创建简单用户界面

Android 应用的图形用户界面是由 View 和 ViewGroup 对象组成的层次结构创建的。View 对象通常是一个 UI 部件如 button 或 text field，ViewGroup 对象是一个不可见的视图容器，它定义了子视图如何布局，如 grid 或 vertical list。

Android 提供了一个 XML 词汇表，对应 View 和 ViewGroup 的子类，你可以在 XML 中使用 UI 元素的层次结构定义自己的用户界面。



图解 ViewGroup 对象如何在布局中组织分支以及包含其它 View 对象

在这节课中，你将创建一个包含 button 和 text field 的 XML 布局。在随后的课程中，你将响应按纽按下时，把 text field 中的内容发送到另一个 Activity。

创建线性布局

从 res/layout/目录中打开 activity_main.xml 文件。

注意：*在 Eclipse 中，你打开一个布局文件时，首先看到的是图形布局编辑器。这是一个使用“所见即所得”工具帮助你创建布局的编辑器。在这节课中，你要直接处理 XML，因此点击屏幕下方的 activity_main.xml 标签打开 XML 编辑器。*

你在创建项目时选择的 BlankActivity 模板包含了一个带相对布局的根视图和 TextView 的子视图的 activity_main.xml 文件。

首先，删除<TextView>元素并把<RelativeLayout>元素改为<LinearLayout>，然后添加 android:orientation 属性并把它的值设为“horizontal”，结果看来如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
</LinearLayout>
```

LinearLayout 是一个视图组（ViewGroup 的子类），它按照 android:orientation 指定的值在垂直或水平方向上排列子视图。LinearLayout 的每一个子视图按照它在 XML 中的顺序出现在屏幕上。

另外两个属性，android:layout_width 和 android:layout_height 用来为所有视图指定大小。

因为 LinearLayout 是布局的根视图，它将填充整个屏幕区域，应用的高度和宽度被设定为“match_parent”，这个值表示视图将会扩展它的宽度和高度以匹配父视图的宽度和高度。

要了解有关布局属性的更多信息，请参见布局指南。

添加 Text Field

要创建一个用户可编辑的文本域，在<LinearLayout>中插入一个<EditText>元素。

像每个 View 对象一样，你必须定义某些 XML 属性以指定 EditText 对象的属性。下面是如何在<LinearLayout>内部声明 EditText 对象：

```
<EditText android:id="@+id/edit_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="@string/edit_message" />
```

有关这些属性：

android:id

它为视图提供一个唯一标识符，以便我们在代码中引用它，例如读取或操作视图对象（我们将下一节课中了解这些）。

@符号用来从 XML 文件中引用任何资源对象，它后面是资源类型（本例中是 id 类型），斜杠，然后是资源名称（本例是 edit_message）。

资源类型前面的加号（+）只有在你首次定义资源 ID 时才需要它，当应用编译时，SDK 工具会在项目的 gen/R.java 文件中使用 ID 名称创建一个新的资源 ID 以引用 EditText 元素，一旦通过这种方式声明了资源 ID，其它对这个 ID 的引用都不再需要加号。只有在指定新资源 ID 时才需要加号，引用具体的资源，如字符串或布局，则不需要加号，详情请见**关于资源对象**。

关于资源对象

资源对象使用一个唯一整数和应用的资源关联起来，比如位图、布局文件或字符串。

每一个资源都在项目的 gen/R.java 文件中定义了一个对应的资源对象。你可以使用类 R 中的对象名引用资源，例如当你需要为 android:hint 属性指定一个字符串值时。你也可以通过 android:id 属性创建一个资源 ID 来和一个视图关联起来，这可以使你在代码中引用这个视图。

每次编译应用时，SDK 工具都会生成 R.java 文件，永远不要手动修改这份文件。

要获取更多信息，请参考“**提供资源**”指南。

android:layout_width 和 android:layout_height

没有使用特定大小的高度和宽度，而是用“wrap_content”做为宽高的值，意味着视图大小会自动匹配内容。如果用“match_parent”代替，那么 EditText 元素将会填充整个屏幕，因为它需要匹配父容器 LinearLayout 的大小。

android:hint

当文本域的内容为空时显示的默认字符串。使用“@string/edit_message”来引用一个在单独的文件中定义的字符串资源，而不是直接使用硬编码的字符串。因为这是引用具体的资源（不是定义标识符），所以不需要加号。由于我们还没有定义字符串，你将会看到一个编译错误。我们会在下一节通过定义字符串来修正它。

注意：*这个字符串资源和元素 ID 同名：edit_message，因为引用资源受资源类型（如 id 或 string）限制，所以使用相同名称不会产生冲突。*

添加字符串资源

当你需要在用户界面中添加文本时，你应该为每个字符串指定一个资源。字符串资源允许你在一个地方管理所有的 UI 文本，方便我们查找和更新文本。外部化字符串还能在你本地化应用以支持不同的语言时，为你提供可替换的字符串。

默认情况下，你的 Android 项目包含了一个字符串资源文件：res/values/strings.xml。我们在文件中添加一个新字符串 edit_message，把它的值设为“Enter a message”(你可以删

除”Hello World”字符串)。

在这份文件中，我们再为后面要添加的按钮加一个字符串“Send”，取名为“button_send”。

最终的 string.xml 如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My First App</string>
    <string name="edit_message">Enter a message</string>
    <string name="button_send">Send</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

要获得本地化应用以支持其它语言的相关信息，参看**支持不同设备**课程。

添加 Button

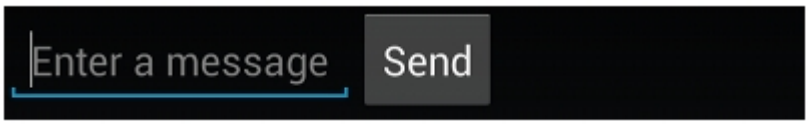
现在在布局中添加一个<Button>，紧随在<EditText>元素之后。

```
<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
```

因为高宽值设置为“wrap_content”，按纽的大小将会自动匹配按纽文本。这个按纽不需要 android:id 属性，因为代码中没有引用它。

让输入框填充整个屏幕宽度

当前的布局把 EditText 和 Button 都设计为自动匹配内容大小，如下图所示：



这对按纽没问题，但对文本域就有点不太合适。因为用户可能会输入比较长的内容，最好能让文本域填充屏幕上剩余的空间。你可以通过使用 android:layout_weight 来指定 LinearLayout 中的 weight 属性做到这一点。

weight 的值是一个数字，它指定了视图使用屏幕剩余空间的数量，这个数量是相对于其它同级视图使用的数量。这有点象饮料配方：2 份伏特加，1 份咖啡甜酒，意思是一份饮料中，伏特加占了三分之二。例如，你指定一个视图的 weight 是 2，另一个视图的 weight 是 1，总和是 3，那么第一个视图将填充剩余空间的三分之二，第二个视图填充余下的部分。如果再添加第三个视图并且 weight 的值为 1，那么第一个视图（weight 值为 2 的那个）将获得剩余空间的一半，另外两个视图每个获得 1/4。

任何视图 weight 的默认值都是 0，所以如果你只为一个视图的 weight 值指定大于 0 的数，那么这个视图将填充屏幕上去掉其它视图所占空间后的剩余空间。因此，要让 EditText 填充屏幕上的剩余空间，我们只需把它的 weight 值设为 1 并且不去修改 Button 的 weight 值。

```
<EditText
        android:layout_weight="1"
        ... />
```

当指定了 weight 的值之后，为了提高布局性能，你应该把 EditText 的 width 值改为零

（0dp）。把宽度设为 0dp 有助于改善布局性能，因为设为”wrap_content”，系统需要计算所需宽度，而设定了 weight 的值后，这种计算是没有意义的，系统最终要根据其它视图的宽度来计算剩余空间的宽度。

下图为重新分配了 weight 值之后的效果：

以下是完成后的布局文件：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```

当你创建项目时，SDK 工具就把这个布局应用到默认的 Activity 类了，因此你可以运行应用察看结果：

1. 在 Eclipse 中，从工具栏中点击 **Run** 。
2. 在命令行中，把路径改变到 Android 项目的根目标中并执行：

```
ant debug
adb install bin/MyFirstApp-debug.apk
```

继续下一节课，学习如何响应按纽按下，从文本域从读取内容，启动另一个 Activity 等更多内容。

启动另一个 Activity

完成了上一节课的内容后，你已经有了一个可以显示包含文本域和按纽的单个 Activity 的应用。在这节课中，你将向 MainActivity 中添加一些代码，使用户点击“发送“按纽时启动一个新的 Activity。

响应“发送”按纽

要响应按纽的点击事件，打开 activity_main.xml 布局文件，并向<Button>元素添加 android:onClick 属性。

```
<Button
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

android:onClick 属性的值"sendMessage"是当用户点击按纽时，系统调用的 Activity 类中的方法名称。

打开 MainActivity 类(在项目的 src/目录下)，添加相应的方法：

```
/** Called when the user clicks the Send button */
public void sendMessage(View view) {
    // Do something in response to button
}
```

这需要你导入 View 类：

```
import android.view.View;
```

提示：在 Eclipse 中，按 Ctrl + Shift + O 导入缺少的类(在 Mac 中按 Cmd + Shift + O)。

要让系统匹配在 android:onClick 属性中指定的方法名，方法签名必须如下所示，具体地说，方法必须：

1. 是 public 的
2. 没有返回值
3. 仅有一个 View 类型的参数（表示被点击的 View 对象）

下一步，我们要完成这个方法以读取文本域的内容并把它传递到另一个 Activity 中。

创建 Intent

Intent 是一个提供运行时绑定两个独立组件（例如两个 Activity）的对象。Intent 表示应用“想要做点什么”，你可以用它来完成各种任务，但最常用的是用来启动另一个 Activity。

在 sendMessage()方法中，创建一个 Intent 对象启动 Activity 被称为 DisplayMessageActivity。

```
Intent intent = new Intent(this, DisplayMessageActivity.class);
```

这里使用的构造函数接受两个参数：

- 第一个参数是一个 Context 对象（这里使用了 this，因为 Activity 类是 Context 类的子类）。
- 系统要传递给 Intent 对象的应用组件的类（本例中，DisplayMessageActivity 引用的 Activity 将被启动）。

  注意：*如果你使用 Eclipse 这样的 IDE，DisplayMessageActivity 将引发一个错误，因为类还不存在。目前先忽略这个错误，我们马上就会创建这个类。*

Intent 对象不仅可以启动另一个 Activity，还能将数据打包传送到 Activity。对 sendMessage()方法中，使用 findViewById()获取 EditText 元素并把它的文本值添加到 Intent 对象中。

```
Intent intent = new Intent(this, DisplayMessageActivity.class);
EditText editText = (EditText) findViewById(R.id.edit_message);
String message = editText.getText().toString();
intent.putExtra(EXTRA_MESSAGE, message);
```

注意：*你现在要导入 android.content.Intent 和 android.widget.EditText 这两个类，还要定义 EXTRA_MESSAGE 常数。*

Intent 可以使用键值对的方式携带各种数据类型的，这个数据集合被称为“extras”，putExtra()方法的第一个参数是键，第二个参数是值。

为了在下一个 Activity 中查询 extra 集合的数据，你应该定义一个公用的常数来表示 extra

中的键，因此我们在 MainActivity 类的顶部增加一个 EXTRA_MESSAGE 的声明。

```
public class MainActivity extends Activity {
    public final static String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";
    ...
}
```

使用应用的包名做前缀来定义你的 Intent 对象的 extra 键是一个好的习惯，这可以确保在你的应用要和其它应用交互时，这个键是唯一的。

启动第二个 Activity

要启动一个 Activity，调用 startActivity()方法并把 Intent 对象传递给它。系统将接受到调用并启动 Intent 对象中指定的 Activity 的实例。

用这些新代码完成 sendMessage()方法，现在它看上去应该是这样：

```
public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.edit_message);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

现在你需要创建 DisplayMessageActivity 类了。

创建第二个 Activity

用 Eclipse 创建新的 Activity：

1. 点击工具栏上的 **New**  。
2. 在接下来的窗口中，打开 **Android** 文件夹，选择 **Android Activity**，点击 **Next**。
3. 选择 BlankActivity，点击 Next。
4. 填写 Activity 详细信息：

   Project：MyFirstApp

   Activity Name：DisplayMessageActivity

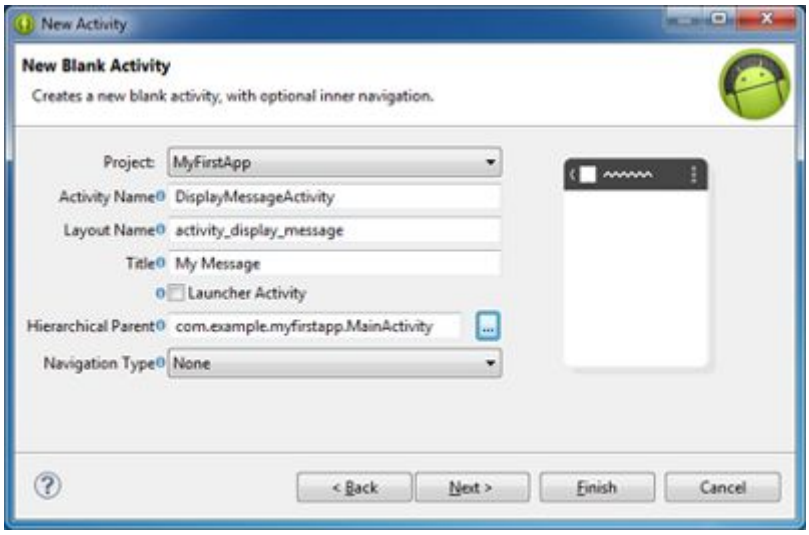
   Layout Name：activity_display_message

   Title：My Message

   Hierarchial Parent：com.example.myfirstapp.MainActivity

   Navigation Type：None

点击 **Finish**。



Eclipse 中的新 Activity 向导

如果你正在使用其它 IDE 和命令行工具，到项目的 src/目录中创建一个新文件命名为 DisplayMessageActivity.java。

打开 DisplayMessageActivity.java 文件，如果你是用 Eclipse 创建的这个 Activity：

- 类已经包含了 onCreate()方法的实现
- 这里还包含了一个 onCreateOptionsMenu()方法的实现，但是在这个应用中不需要它，你可以删除它。
- 这里还包含了一个 onOptionsItemSelected()方法的实现，它用来处理操作栏中“Up”的行为，保留它。

因为 ActionBar API 仅在 HONEYCOMB（API 级别 11）或更高版本中支持，你需要在 getActionBar()方法外面添加一个条件语句来检查当前平台版本。附带地，你还必须在 onCreate()方法前添加@SuppressLint(“NewApi”)标记，以避免 Lint 错误。

DisplayMessageActivity 类看起来如下所示：

```
public class DisplayMessageActivity extends Activity {

    @SuppressLint("NewApi")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_message);

        // Make sure we're running on Honeycomb or higher to use ActionBar APIs
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
            // Show the Up button in the action bar.
            getActionBar().setDisplayHomeAsUpEnabled(true);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
```

```
            case android.R.id.home:
                NavUtils.navigateUpFromSameTask(this);
                return true;
            }
            return super.onOptionsItemSelected(item);
        }
    }
```

如果你使用了 Eclipse 以外的 IDE，用上面的代码更新你的 DisplayMessageActivity 类。

Activity 的任何子类都必须实现 onCreate()方法。当创建 Activity 新实例时由系统调用这个方法。你必须这个方法中用 setContentView()声明 Activity 布局，并对 Activity 组件进行初始化。

注意：*如果你使用 Eclipse 之外的 IDE，你的项目中不会包含 setContentView()方法所需的 activity_display_message 布局文件，这没关系，稍后你将更新 onCreate()方法，不会用到那个布局文件。*

添加标题字符串

如果你使用的是 Eclipse，你可以跳到下一小节，因为 Eclipse 已经自动为新 Activity 提供了标题字符串。

如果你使用的是 Eclipse 之外的 IDE，在 strings.xml 文件中为新 Activity 添加标题字符串。

```
<resources>
    ...
    <string name="title_activity_display_message">My Message</string>
</resources>
```

添加到清单文件中

所有的 Activity 都必须在清单文件 AndroidMainfest.xml 中用<Activity>元素声明。

当你使用 Eclipse 工具创建 Activity 时，它会创建默认的条目。如果你使用的是其它 IDE，你需要手动在清单文件中添加条目，它看来如下所示：

```
<application ... >
    ...
    <activity
        android:name="com.example.myfirstapp.DisplayMessageActivity"
        android:label="@string/title_activity_display_message"
        android:parentActivityName="com.example.myfirstapp.MainActivity" >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.myfirstapp.MainActivity" />
    </activity>
</application>
```

android:parentActivityName 定义了这个 Activity 在应用的逻辑层次中的上级 Activity。系统使用这个值来实现默认的导航行为，例如 Android4.1 或更高版本中的“向上导航”。你可以使用支持库并添加像这里的<meta-data>元素那样，为 Android 旧版本提供同样的导航行为。

注意：*你的 Android SDK 已经包含了最新的 Android 支持库，它包含在 ADT 包中。如果你使用不同的 IDE，你应该在“添加平台和包”这个步骤中安装它。当使用 Eclipse 中的模板时，支持库会自动添加到你的应用项目中（你可以在 Android Depenencies 的列表中看到库的 JAR 文件）。如果你使用的不是 Eclipse，你需要手动把库添加的你的项目中。请参考“设置支持库”指南。*

如果你正在使用 Eclipse 开发，你现在就可以运行应用，不过不要期望过高。点击“发送”按纽启动第二个 Activity，呈现的是默认的“Hello World”模板提供的布局。你马上会更新这个 Activity，显示一个自定义的文本视图来替换它，因此如果你用的不是 Eclipse，别担心，应用还没有编译。

接收 Intent

每个 Activity 都由一个 Intent 调用，无论用户如何导航，你都可以通过调用 getIntent()方法获得启动 Activity 的 Intent 对象并且接收它包含的数据。

在 DisplayMessageActivity 类的 onCreate()方法中，获取 Intent 对象和 MainActivity 传递过来的消息文本。

```
Intent intent = getIntent();
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

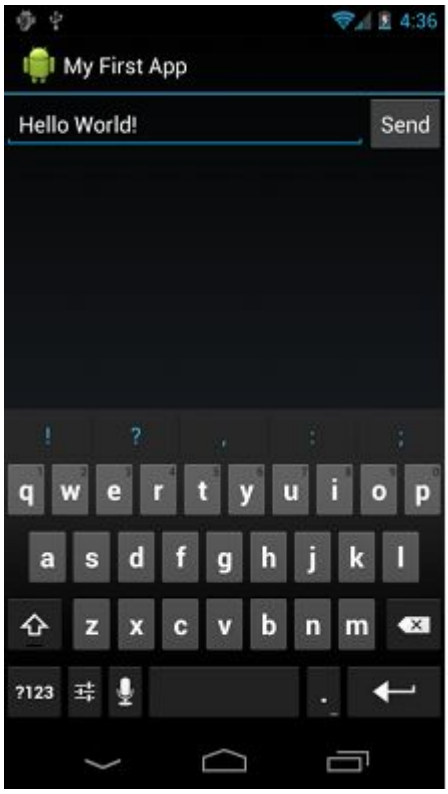
显示消息文本

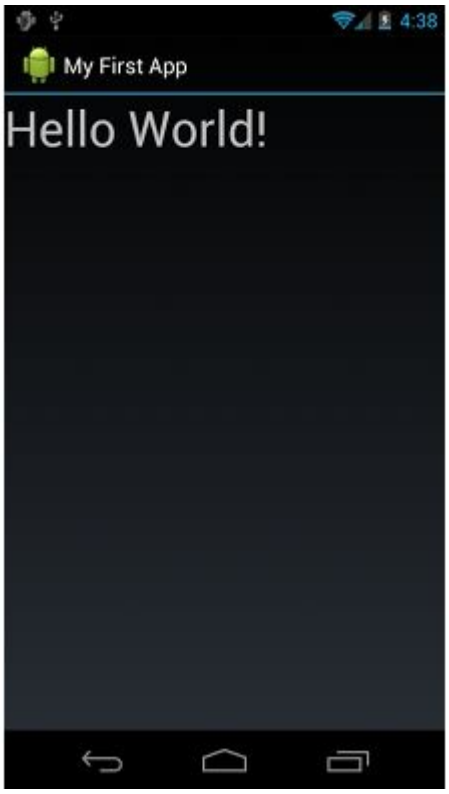
在屏幕上显示消息文本，创建一个 TextView 部件并且使用 setText()方法来设置文本，然后通过 setContentView()把它添加为 Activity 布局的根视图。

DisplayMessageActivity 类的 onCreate()方法完整的代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Get the message from the intent
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

    // Create the text view
    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(message);

    // Set the text view as the activity layout
    setContentView(textView);
}
```

现在你可以运行应用了，当它打开后，在文本域中键入消息，点击 Send，消息会显示在第二个 Activity 中。





就是这样，你创建了你第一个 Android 应用!
要学习更多内容，跟我们进入下一课吧。